Verifiable Properties of Database Transactions

Michael Benedikt, Timothy Griffin, and Leonid Libkin

Copyright © 1996 by the Association for Computing Machinery, Inc. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Publications Dept, ACM Inc., fax +1 (212) 869-0481, or `permissions@acm.org`.

Verifiable Properties of Database Transactions

Michael Benedikt **Timothy Griffin** **Leonid Libkin**
Bell Laboratories
600 Mountain Avenue, Murray Hill NJ 07974, USA
e-mail: {benedikt, griffin, libkin}@research.att.com

Abstract

It is often necessary to ensure that database transactions preserve integrity constraints that specify valid database states. While it is possible to monitor for violations of constraints at run-time, rolling back transactions when violations are detected, it is preferable to verify correctness statically, *before* transactions are executed. This can be accomplished if we can verify transaction safety with respect to a set of constraints by means of calculating *weakest preconditions*. We study properties of weakest preconditions for a number of transaction and specification languages. We show that some simple transactions do not admit weakest preconditions over first-order logic. We also show that the class of transactions that admit weakest preconditions over first-order logic cannot be captured by any transaction language. We consider a strong local form of verifiability, and show that it is different from the general form. We define robustly verifiable transactions as those that can be statically analyzed regardless of extensions to the signature of the specification language, and we show that the class of robustly verifiable transactions over first order logic is exactly the class of transactions that admit the local form of verifiability. We discuss the implications of these results for design of verifiable transaction languages.

1 Introduction

Databases are typically required to satisfy a collection of *integrity constraints* — sentences that specify the valid states of the database. Database transactions that update the database must preserve these constraints. One approach to the integrity maintenance problem is to defer the detection of potential violations of integrity constraints until run-time, thus preventing the violation of integrity that may be caused by transactions. However, this approach results in potentially expensive roll-back operations. In order to avoid this, one can attempt to verify *statically*, before a transaction is executed, that it will necessarily maintain all constraints. If the result of this verification shows that the transaction may sometimes fail to maintain some of the constraints, then modifications can be made until correctness is achieved.

If T is a transaction and α is a constraint, we say that T preserves α if for every database D:

$$D \models \alpha \text{ implies } T(D) \models \alpha.$$

We use the notation $D \models \alpha$ to mean that α is true for the database D.

According to the approach described above, we would like to be able to check transactions to see if they will always preserve α. But, as the following result tells us, one cannot hope to have an algorithm for checking this, even for simple transactions and simple constraint specification languages.

Fact A (cf. [2]) *For transactions specified as select-project-join expressions of the relational algebra, and integrity constraints specified as first-order sentences, it is undecidable to check if a given transaction preserves a given integrity constraint.*

Similar results hold for transactions that consist of simple updates and deletions. Therefore, any approach to automatic verification of transaction safety (such as the approach of [31, 32] that uses theorem-provers) is inherently limited. Instead, we adopt an approach based on modifying transactions to ensure safety.

Given a transaction T and a constraint α, we would like to find another constraint β that is satisfied by a database D *before* T is committed if and only if α is satisfied *afterwards*. In other words, for every database D:

$$D \models \beta \text{ if and only if } T(D) \models \alpha.$$

We shall call such a β a *weakest precondition* for α, with respect to T, and denote it by $\mathsf{wpc}(T, \alpha)$. Since we consider only transactions that do not diverge, this notion of weakest precondition is equivalent to the classical one of Dijkstra [11, 12].

The existence of weakest preconditions would not contradict Fact A, as we would not be able to determine the validity of the weakest precondition. At the same time it would solve the problem of statically verifying database integrity since any transaction T could be transformed to

$$\underline{\mathsf{if}}\ \mathsf{wpc}(T, \alpha)\ \underline{\mathsf{then}}\ T\ \underline{\mathsf{else}}\ \ abort$$

which will maintain consistency while avoiding any rollback operations. Furthermore, we could then modify the resulting transaction by applying simplification algorithms to $\mathsf{wpc}(T, \alpha)$, thus recapturing many of the benefits of approaches based on validity-checking. This is the fundamental idea underlying many algorithms for the automatic maintenance of integrity constraints [26, 17, 18, 27, 25].

Given the attractiveness of this approach to integrity maintenance, it is important to understand the tradeoffs involved in designing transaction and specification languages with respect to our ability to express weakest preconditions. In this paper, we will concentrate on the basic principles behind developing safe transactions via preconditions, rather than specific algorithms that attempt to detect violations of integrity.

The main goal of this paper is to study the properties of weakest preconditions for a variety of transaction and specification languages. In particular, we concentrate on specification languages that are relevant to integrity constraints, such as first-order logic over the database schema. Below we give a brief overview of the main results of the paper.

Assume that we have two languages $\mathcal{L}_1$ and $\mathcal{L}_2$ in which constraints may be specified. Also assume that we have a transaction language $\mathcal{TL}$. We say that $\mathcal{TL}$ is $\mathcal{L}_1$-*verifiable over* $\mathcal{L}_2$ iff for every $T \in \mathcal{TL}$ and every $\alpha \in \mathcal{L}_2$, it is the case that a weakest precondition for α with respect to T is in $\mathcal{L}_1$. If $\mathcal{L} = \mathcal{L}_1 = \mathcal{L}_2$, we speak of $\mathcal{L}$-*verifiable* transaction languages. Similar concepts have been studied in the more general programming language context, see [6, 9].

Verifiable transaction languages are exactly those for which the weakest-precondition approach to integrity maintenance is feasible. An example of a language verifiable over first-order logic is the first-order transaction language of [28]. However, no results exist for transaction languages that possess more power than first-order logic (for example, for languages that allow some forms of recursion). The first result of this paper is negative: it shows that adding recursion destroys first-order verifiability.

Theorem B *Let $\mathcal{TL}$ be a transaction language that expresses at least one of the following:*

- *transitive closure;*
- *deterministic transitive closure;*
- *same-generation (even restricted to trees).*

Then $\mathcal{TL}$ is not verifiable over first-order logic.

We shall also prove a number of extensions of this result. For example, any language that can express all of the relational calculus plus (deterministic) transitive closure is not monadic Σ^1_1-verifiable over first-order logic, nor is it verifiable over first-order logic over a signature that contains a set of interpreted operations on the underlying domain, or over first-order logic supplemented by a generalized quantifier for equal cardinality.

We next fix our specification language to be first-order logic, and look at the set $\mathcal{WPC}(\mathrm{FO})$ of transactions that have weakest preconditions for first-order sentences. We show that this collection cannot be captured by any transaction language.

Theorem C *There is no transaction language that expresses exactly all the transactions from $\mathcal{WPC}(\mathrm{FO})$. This remains true if* FO *is replaced by first-order logic over a signature containing a recursive collection of interpreted recursive functions and predicates.*

The ability to statically check global conditions — that is, conditions that must be satisfied by the database as a whole — is, as noted above, extremely helpful. However, it is often desirable to be able to statically check local conditions on individual tuples as well. In particular, one often would like to find a condition that must be satisfied by a database D and an object t (typically, a record) in order for t to belong to $T(D)$. It is natural to compare the class $\mathcal{WPC}(\mathcal{L})$ with the class of transactions for which this *local* checking can be carried out. For a specification language $\mathcal{L}$, the natural definition of such a class is the set of transactions admitting *prerelations.* Informally, a transaction T admits prerelations if for every relation R of arity n there exists a formula $\mathsf{pre}^T_R(x_1, \ldots, x_n)$ such that, for every n-tuple $t = (d_1, \ldots, d_n)$ and every database D:

$$\begin{array}{ll} & D \models \mathsf{pre}^T_R(d_1, \ldots, d_n) \ \text{ and } \ d_1, \ldots, d_n \in \Gamma(D) \\ \Leftrightarrow & T(D) \models R(d_1, \ldots, d_n) \end{array}$$

Here $\Gamma(D)$ is a finite set that extends the active domain of the database. The precise definition will be given in the next section.

The condition $T(D) \models R(d_1, \ldots, d_n)$ means that t belongs to the relation R after the transaction T is applied to the database D. Thus, prerelations allow us to test membership before transactions are committed.

In papers like [28] prerelations are used in place of weakest preconditions. The question of the relationship of these two notions, however, has not been addressed. We shall see that for most specification languages of interest, it is the case that any transaction that admits prerelations also admits weakest preconditions. The converse, however, is not true even in very simple settings.

Theorem D *There exists a transaction T in $\mathcal{WPC}$(FO) that does not admit prerelations. Moreover, T can be chosen to be* Datalog$^{\neg}$*-definable.*

However, this result is not robust: if FO is extended by adding a constant for each element of the domain, then the transaction from Theorem D no longer provides a separation example. Motivated by this, we say that a transaction language $\mathcal{TL}$ is *robustly verifiable* over a first-order signature Ω if, for any extension of Ω to a signature Ω' via a set of recursive predicates and functions, all transactions in $\mathcal{TL}$ remain verifiable over the corresponding extension of first-order logic. We then prove a positive result that characterizes maximal robustly verifiable languages. Let FO_c be first-order logic over the relational schema supplemented with symbols for all constants.

Theorem E *There exists a transaction language that captures the class of transactions admitting prerelations over* FO_c*. Moreover, this language is the maximal robustly verifiable language over* FO_c*. In particular, every transaction robustly verifiable over* FO_c *is equivalent to a transaction that admits prerelations.*

This has implications for transaction language design. If we are interested in a "nice" transaction language that is verifiable in an extensible way over all first-order languages, then we cannot hope that the language will be more expressive than the first-order transaction language defined in [28, 29].

Organization We rigorously define our formal setting in the next section. In section 3 we study the limitations of weakest preconditions. Section 4 studies the relationship between weakest preconditions and prerelations. Robustly verifiable transaction languages are the subject of section 5. Concluding remarks are given in section 6.

In this extended abstract we only give sketches of the proofs; details can be found in [8].

2 Formal setting

We fix the domain to be a countably infinite set $\mathcal{U}$. A *relational schema* is a nonempty set $SC = \langle R_1, \ldots, R_k \rangle$ of predicates, with a finite arity $n_i > 0$ associated with each R_i. A *database* over SC is an interpretation of each R_i as a finite subset of $\mathcal{U}^{n_i}$. Most often we will use the schema that consists of a single binary predicate E. Then databases are interpreted as finite graphs whose nodes are elements of $\mathcal{U}$.

We shall use two notions – *specification language*, in which constraints are specified, and *transaction language*. A specification language $\mathcal{L}$ is a recursive subset of a set of strings with an associate subset $sent(\mathcal{L})$ of *sentences*, which is also recursive. As stated in the introduction, we wish to focus on the specification languages that are relevant to the integrity constraints. In particular, we will study

- FO – (pure) first-order logic (that is, first-order logic over the relational schema SC).
- FO_c — first-order logic over the signature that consists of SC supplemented with constant symbols for all elements of $\mathcal{U}$.
- $\mathrm{FO}_c(\Omega)$ — first-order logic over the signature that consists of SC supplemented with constant symbols as above and a recursive collection Ω of recursive functions and predicates over $\mathcal{U}$.
- Monadic Σ^1_1, whose formulae are of the form $\exists A_1 \ldots \exists A_k.\Psi$ where A_i's are monadic (unary) predicates and Ψ is a first-order formula over $SC \cup \langle A_1, \ldots, A_k \rangle$.

For all specification languages it will be the case that we shall have a binary relation $\models$ for validity between databases and sentences of the languages. For all the languages above, validity is defined in the usual way.

A transaction language consists of 1) Syntax: a recursive subset of the set S of strings over a finite alphabet, and 2) Semantics: a total recursive function M that takes a string s, and an encoding of a database, and returns either another database encoding or "error", provided $s \in S$. Examples include relational algebra and calculus, SQL, first-order transaction language of [28], a variety of languages from [3, 4] and so on. We shall always use the same symbol for both syntactic and semantic objects. For example, for the language given by $\langle S, M\rangle$, $T \in S$ and a database D, we will write $T(D)$ to denote the result of M on T and the encoding of D.

We are interested in maintaining database integrity. To formalize this problem, let $\mathcal{L}$ be a specification language and $\mathcal{TL}$ a transaction language. We define the following problem:

> PRESERVE($\mathcal{TL}, \mathcal{L}$): Given $T \in \mathcal{TL}$ and α in $\mathcal{L}$, is it the case that $D \models \alpha$ implies $T(D) \models \alpha$ for every database D?

According to Fact A, this problem is undecidable even for transactions specified by the simplest form of SQL statements and for first-order constraints. This can be derived from a result of [2].

Proposition 1 *Let $\mathcal{L}$ be* FO*, and let $\mathcal{TL}$ contain the transactions given by the select-project-join expressions of the relational algebra. Then* PRESERVE($\mathcal{TL}, \mathcal{L}$) *is undecidable.* □

As we explained in the introduction, a way around this is to introduce weakest preconditions. Below is the main definition.

Definition *1) Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two specification language. We say that a transaction T (a total map from databases to databases) has* $\mathcal{L}_1$-weakest preconditions *over $\mathcal{L}_2$ if for every $\mathcal{L}_2$-sentence α there exists an $\mathcal{L}_1$ sentence $\mathsf{wpc}(T, \alpha)$, computable from T and α, such that for every database D:*

$$D \models \mathsf{wpc}(T, \alpha) \quad \textit{if and only if} \quad T(D) \models \alpha.$$

The class of transactions having $\mathcal{L}_1$-weakest preconditions over $\mathcal{L}_2$ is denoted by $\mathcal{WPC}(\mathcal{L}_1, \mathcal{L}_2)$. We also write $\mathcal{WPC}(\mathcal{L})$ for $\mathcal{WPC}(\mathcal{L}, \mathcal{L})$.

2) We say that a transaction language $\mathcal{TL}$ is $\mathcal{L}_1$-verifiable *over $\mathcal{L}_2$ if every $\mathcal{TL}$ transaction T is in $\mathcal{WPC}(\mathcal{L}_1, \mathcal{L}_2)$. If $\mathcal{L} = \mathcal{L}_1 = \mathcal{L}_2$, we speak of a language verifiable over $\mathcal{L}$.*

Proposition 2 *(a) Let $\mathcal{L}_1' \preccurlyeq \mathcal{L}_1$ and $\mathcal{L}_2 \preccurlyeq \mathcal{L}_2'$. Then $\mathcal{WPC}(\mathcal{L}_1', \mathcal{L}_2') \subseteq \mathcal{WPC}(\mathcal{L}_1, \mathcal{L}_2)$.*
(b) It is possible to find languages $\mathcal{L} \preccurlyeq \mathcal{L}'$ and a transaction T such that $T \in \mathcal{WPC}(\mathcal{L}) - \mathcal{WPC}(\mathcal{L}')$.
(c) It is possible to find languages $\mathcal{L} \preccurlyeq \mathcal{L}'$ and a transaction T such that $T \in \mathcal{WPC}(\mathcal{L}') - \mathcal{WPC}(\mathcal{L})$. □

Similar notions have been studied in a more general programming language context. For example, weakest preconditions over first-order logic for a simple while loop language can be expressed in infinitary logic [6] or in weak second-order logic [9]. However, proposition 2 shows that it is not always possible to extend results of this kind to less expressive specification languages. In particular, according to Theorem A, weakest preconditions for transitive closure (which is definable with while loops) are not first-order expressible.

We will also use *prerelations* which allow for testing of local conditions. We are now ready to formalize the definition we presented informally in the introduction. Let Ω be a signature. By TERM(Ω) we denote the set of all Ω-terms which are built up from variables by using the symbols from Ω. (We treat constants as functions of arity zero.) Given a database D, its (active) *domain* $\mathrm{dom}(D)$ is the set of all elements of $\mathcal{U}$ that occur in the database. For a set $\Gamma \subseteq$ TERM(Ω) and a database D we define $\Gamma(D)$ to be the set of all $x \in \mathcal{U}$ such that $x = \tau(y_1, \ldots, y_n)$ for some n-ary term $\tau \in \Gamma$ and some $y_1, \ldots, y_n \in \mathrm{dom}(D)$.

Definition *Let T be a transaction. We say that T admits* prerelations *over $\mathcal{L}$ if there exists a finite collection of terms Γ and a collection of $\mathcal{L}$-formulae $\mathsf{pre}_1^T, \ldots, \mathsf{pre}_k^T$, pre_i^T having n_i free variables, such that for any R_i in the schema and any $d_1, \ldots, d_{n_i} \in \mathcal{U}$, it is the case the for every database D:*

$$\begin{aligned} & D \models \mathsf{pre}_i^T(d_1, \ldots, d_{n_i}) \textit{ and } d_1, \ldots, d_{n_i} \in \Gamma(D) \\ \Leftrightarrow\ & T(D) \models R_i(d_1, \ldots, d_{n_i}) \end{aligned}$$

The collection of all transactions that admit prerelations over $\mathcal{L}$ is denoted by $\mathcal{PR}(\mathcal{L})$.

The intuition behind the definition above is this. Since $T(D) \models R_i(d_1, \ldots, d_{n_i})$ says that the tuple $(d_1, \ldots, d_{n_i})$ belongs to R_i after T is applied, using prerelations we can test, *before* T is committed, whether a given tuple will be in the database afterwards. Thus prerelations can be viewed as providing the ability to verify local properties of databases as well as global properties. Prerelations have been used extensively in [28]. If $\mathcal{L}$ is pure first-order logic (in particular, Ω is empty and the only terms are variables), then the notion of having prerelations reduces to the familiar notion of a first-order definable transaction.

For all the languages we shall consider in this paper it is the case that $\mathcal{PR}(\mathcal{L}) \subseteq \mathcal{WPC}(\mathcal{L})$. Indeed, to construct $\mathsf{wpc}(T, \alpha)$ for any $T \in \mathcal{PR}(\mathcal{L})$ we can use Γ and pre_i^T to simulate the new database state after the transaction, and then substitute all symbols for R_i in α by the formulae defining the new state. It is easy to see that the power of first-order logic is sufficient for doing this. Later in the paper we examine this inclusion and show that for most languages of interest it is strict.

3 Weakest preconditions

The goal of this section is to study weakest preconditions for first-order logic and its extensions. We first prove that a language that can express (deterministic) transitive closure or the same-generation query cannot have weakest preconditions over FO. We then extend this result to more powerful specification languages, and finally prove that no transaction language captures the class of transactions having weakest preconditions over first-order logic.

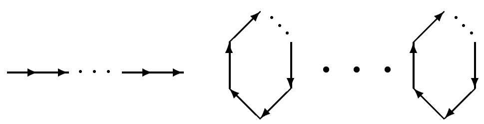
We need a few technical definitions. Given a graph $G = \langle V, E \rangle$, its *deterministic transitive closure* (cf. [20]), $dtc(G)$, is the graph $\langle V, E' \rangle$ where $(x,y) \in E'$ iff $(x,y) \in E$ or there is a sequence of nodes $x = x_1, x_2, \ldots, x_n = y$ such that each $(x_i, x_{i+1}) \in E$, and furthermore, each x_i has out-degree 1, $i = 1, \ldots, n-1$. The transitive closure of a graph is denoted by $tc(G)$.

Given a graph $G = \langle V, E \rangle$, the *same-generation* query returns the graph $sg(G) = \langle V, E' \rangle$ where $(x,y) \in E'$ iff there is a node $v \in V$ and two walks in G from v to x and from v to y that have the same length. Usually the same-generation query is asked for trees. So we define the class of queries

$$\mathcal{SG}_{tree} \;=\; \{q \mid \text{ if } G \text{ is a tree, then } q(G) = sg(G)\}$$

In particular, $sg \in \mathcal{SG}_{tree}$.

A *chain* is a graph of the form $\{(x_1, x_2), \ldots, (x_{n-1}, x_n)\}$ where all x_is are distinct. A *simple cycle* is a graph of the form $\{(x_1, x_2), \ldots, (x_{n-1}, x_n), (x_n, x_1)\}$ where all x_is are distinct. A *chain-and-cycle graph*, or a *C&C-graph* consists of $n \geq 1$ connected components, of which exactly one is a chain, and others (perhaps zero) are simple cycles. An example of a C&C-graph is shown below.



Lemma 1 *The class of C&C-graphs is first-order definable. Neither chains nor cycles are first-order definable.*

Proof. A graph is a C&C-graph iff it has exactly one root (node with in-degree zero), exactly one leaf (node with out-degree zero), the root has out-degree 1, the leaf has in-degree one, and all other nodes have both in- and out-degrees 1. All these conditions are first-order definable. That chains and cycles are not first-order definable can be proved straightforwardly using standard techniques (cf. [15, 23]). $\Box$

For the rest of the paper $\Psi_{\mathrm{c\&c}}$ denotes a first-order sentence in the language of $E(\cdot,\cdot)$ that defines C&C-graphs.

3.1 Languages not verifiable over FO

In this subsection we prove Theorem B from the introduction. That is,

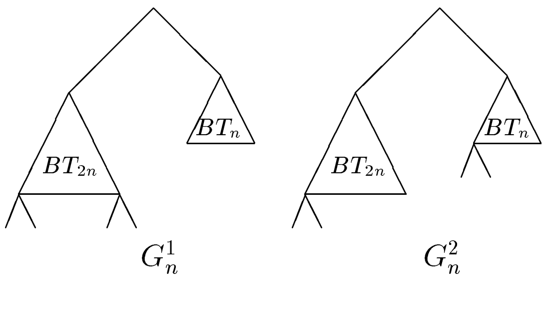
Theorem 2 *Assume that $\mathcal{TL}$ can express one of the following:*

- *transitive closure;*
- *deterministic transitive closure;*
- *any query from $\mathcal{SG}_{tree}$.*

Then $\mathcal{TL}$ is not verifiable over FO.

Proof sketch. If $tc \in \mathcal{WPC}(\mathrm{FO})$, then for $\alpha \equiv \forall x \forall y. E(x,y)$, $\mathsf{wpc}(tc, \alpha)$ is a connectivity test; hence $tc \notin \mathcal{WPC}(\mathrm{FO})$. Assuming $dtc \in \mathcal{WPC}(\mathrm{FO})$, let $\alpha \equiv \forall x \forall y.(x \neq y \rightarrow E(x,y) \vee E(y,x))$. Then $G \models \Psi_{\mathrm{c\&c}} \wedge \mathsf{wpc}(dtc, \alpha)$ iff G is a chain. Since testing for chain is not FO-definable, $dtc \notin \mathcal{WPC}(\mathrm{FO})$.

Let $q \in \mathcal{SG}_{tree}$. Assume that $q \in \mathcal{WPC}(\mathrm{FO})$. For G a tree, $q(G)$ consists of a number of connected components, each of them being a complete graph. Let α be a first-order sentence saying that the number of such connected components of cardinality 4 is exactly two, and let β be $\mathsf{wpc}(q, \alpha)$. Next we define graphs G_n^1 and G_n^2 as shown in the picture below. Here BT_k stands for a binary tree in which all paths from the root to the leaves have the same length k. In G_n^1, the only children added to the tree of length $2n$ are the children of two nodes for which the root is the only common ancestor.



G_n^1 G_n^2

Now observe that for every $n \geq 1$ we have $G_n^1 \models \beta$ and $G_n^2 \models \neg\beta$. Thus, β is a first-order sentence that distinguishes the families $\mathcal{G}^1 = \{G_n^1 \mid n \geq 1\}$ and $\mathcal{G}^2 = \{G_n^2 \mid n \geq 1\}$. However, for any first-order sentence ψ, there exists a number N such that for all $n > N$, $G_n^1 \models \psi$ iff $G_n^2 \models \psi$.

To prove this statement we use Hanf's technique as presented for the finite case in [14]. An r-type of a node a is the isomorphism type of the subgraph given by all the nodes whose distance from a is at most r, with a as a distinguished node. If ψ of quantifier rank k is given, take r to be 3^k. Then for any $n > 2r$, G_n^1 and G_n^2 have the same number of subgraphs with a distinguished node that realize each r-type, and thus, by the result of [14], G_n^1 and G_n^2 agree on ψ. Consequently, no first-order sentence distinguishes $\mathcal{G}^1$ and $\mathcal{G}^2$. This finishes the proof of $q \notin \mathcal{WPC}(\mathrm{FO})$. $\square$

3.2 Weakest preconditions for more powerful logics

Here we extend the result of the previous subsection for a number of more powerful logics. Some of them have been defined already. One more that we study here is FO with the generalized quantifier for equal cardinality $Q_{eq\_card}x.(P(x), P'(x))$ with the semantics $card(\{x \mid P(x)\}) = card(\{x \mid P'(x)\})$. (It is known that equal cardinality is not FO-definable.)

Theorem 3 *Let $\mathcal{L}$ be either* FO + *eq_card, or* $\mathrm{FO}_c(\Omega)$ *for an arbitrary Ω, or monadic Σ_1^1. Then neither transitive closure nor deterministic transitive closure is in $\mathcal{WPC}(\mathcal{L})$.*

Proof sketch. The proof for FO + *eq_card* and $\mathrm{FO}_c(\Omega)$ is essentially the same as the proof of theorem 2, as we have to show that connectivity and testing for chain are not definable in those logics. This follows from the results of [23], [7] and [19]. In the case of monadic Σ_1^1, the proof for *tc* is the same as before, since connectivity is not expressible in monadic Σ_1^1 [5]. For *dtc*, we have to show that testing whether a graph is a chain is not monadic Σ_1^1-definable. This can be done using the result of [14] that for any monadic Σ_1^1 sentence ψ there exists a number n such that ψ can not distinguish one cycle of length n from two cycles of length n. $\square$

Corollary 1 *Let $\mathcal{L}$ be either* FO + *eq_card, or* $\mathrm{FO}_c(\Omega)$ *for an arbitrary Ω, or monadic Σ_1^1. Then neither transitive closure nor deterministic transitive closure is in $\mathcal{WPC}(\mathcal{L}, \mathrm{FO})$.* $\square$

3.3 The structure of transactions with weakest preconditions

Here we prove the result showing that the class of verifiable transactions cannot be captured by a transaction language. (By capturing we mean that a language expresses exactly the transactions from a given set.)

Theorem 4 *There is no transaction language that captures $\mathcal{WPC}(\mathrm{FO})$. Furthermore, for an arbitrary signature Ω, there is no transaction language that captures $\mathcal{WPC}(\mathrm{FO}_c(\Omega))$.*

Proof sketch. We prove the result for $\mathcal{WPC}(\mathrm{FO}_c(\Omega))$ first. Assume without loss of generality that the schema consists of a binary relational symbol $E(\cdot,\cdot)$. The proof below will automatically apply to any relational schema. Given a transaction language $\mathcal{TL}$, we show that it cannot capture $\mathcal{WPC}(\mathrm{FO}_c(\Omega))$. We will assume that the transactions in $\mathcal{TL}$ are enumerated as $\langle T_i\rangle_{i>0}$, graphs as $\langle G_i\rangle_{i>0}$, and sentences of $\mathrm{FO}_c(\Omega)$ as $\phi_0, \phi_1, \ldots$. We define the equivalence relation $=_n$ on graphs by $G =_n G'$ iff $G \models \phi_i \Leftrightarrow G' \models \phi_i$ for all $i \leq n$.

We define a function $H(m,n)$ by letting $H(m,n)$ be lexicographically least pair $\langle i,j\rangle$ such that $m < i < j$ and $G_j =_n G_i$ and $G_j \neq G_i$. Then $P(n)$ and $Q(n)$ are defined inductively by $P(0) = Q(0) = 1$, and by letting $P(n+1)$ be the first component of $H(P(n),n)$, and $Q(n+1)$ be the second component of $H(P(n),n)$.

Finally, define the following transaction T. If $i = 1$ or i is not in the range of P, then $T(G_i) = G_i$. If $i > 1$ is in the range of P, consider the graph $G' = T_{P^{-1}(i)}(G_i)$. Since $G_{P(k)} \neq G_{Q(k)}$ for every $k > 0$, we know that for $j = Q(P^{-1}(i))$ it is the case that $G_i \neq G_j$. Now we define $T(G_i)$ in this case to be the one of G_i, G_j that is not equal to G', and if both are unequal to G', then we define $T(G_i)$ to be $G_{\min(i,j)}$.

The theorem now follows from the following three claims: the transaction T is a total recursive function, it is in $\mathcal{WPC}(\mathrm{FO}_c(\Omega))$, and it is different from each T_i in $\mathcal{TL}$. For $\mathcal{WPC}(\mathrm{FO})$ the proof uses an enumeration of isomorphism classes of graphs instead of an enumeration of graphs. $\square$

As the last result of this section, we show that one cannot find a condition on degrees of nodes in graphs that describes the transactions with weakest preconditions. We are motivated by the following property of first-order queries, established in [23]. For a graph G, its *degree count* $dc(G)$ is the number of different in- and out-degrees of nodes of G. Then, for any first-order query q, $dc(q(G))$ is bounded by a number that depends only on q and the maximal possible in- or out-degree in G. For example, if G is an arbitrary binary tree, then an upper bound on $dc(q(G))$ is fully determined by q.

Ome may ask whether the class $\mathcal{WPC}(\mathrm{FO})$, which includes all first-order definable transactions, satisfies this property. We can show that this is not the case. In fact, a degree-based characterization is impossible for $\mathcal{WPC}(\mathrm{FO})$. Given any function $f : \mathbb{N} \to \mathbb{N}$ such that $f(n) > 0$, let $\mathcal{Q}_f$ be the family of graph queries q such that $dc(q(G)) \leq f(dc(G))$.

Corollary 2 *For any f, both $\mathcal{Q}_f - \mathcal{WPC}(\mathrm{FO})$ and $\mathcal{WPC}(\mathrm{FO}) - \mathcal{Q}_f$ are nonempty.*

Proof. Let q return the diagonal if its input graph is connected, and the complete graph on the input's nodes if it is not. Since connectivity is not first-order, $q \notin \mathcal{WPC}(\mathrm{FO})$. On the other hand, $q \in \mathcal{Q}_{\lambda x.1}$. Conversely, $T \in \mathcal{WPC}(\mathrm{FO})$ that will be described later in the proof of theorem 5 is such that when its input is a chain, it constructs its transitive closure. Hence, T is not in $\mathcal{Q}_f$ for any f. $\square$

4 Preconditions vs. prerelations

The goal of this section is to examine the relationship between $\mathcal{WPC}(\mathrm{FO})$ and $\mathcal{PR}(\mathrm{FO})$. In particular, we find a transaction that separates them. However, we are not interested in just any transaction. Usually transformations expressed by database languages are required to satisfy certain properties. In particular, one is most often interested in transactions and queries computable in polynomial time. In addition, queries and transformations are sometimes required to be *generic*, that is, invariant under any permutation of the universe.

There is a large body of research dealing with languages capable of expressing polynomial-time generic database queries or transformations. Languages in which such transformations can be expressed include datalog with negation [1], languages based on structural recursion [10], loops [24], nondeterministic *reduce* operators [21], *while* queries [1] etc.

Hence, our first goal is to produce a generic polynomial time computable transaction that separates $\mathcal{WPC}(\mathrm{FO})$ from $\mathcal{PR}(\mathrm{FO})$. The transaction exhibited below can be expressed in all languages mentioned in the previous paragraph. The result below provides the proof of Theorem D from the introduction. After proving theorem 5 we proceed to show that $\mathcal{PR}(\mathrm{FO}_c(\Omega))$ can be captured by a transaction language, which gives us a separation result for $\mathcal{WPC}(\mathrm{FO}_c(\Omega))$ and $\mathcal{PR}(\mathrm{FO}_c(\Omega))$. We close the section by showing that $\mathcal{WPC}(\mathrm{FO}_c)$ and $\mathcal{PR}(\mathrm{FO}_c)$ coincide on generic transactions.

Theorem 5 *There exists a polynomial-time computable generic transaction T such that*

$$T \in \mathcal{WPC}(\mathrm{FO}) - \mathcal{PR}(\mathrm{FO}).$$

Proof sketch. Given a C&C-graph G, let *chain*(G) be the connected component of G which is a chain. We consider the following transaction T on graphs $G = \langle X, E\rangle$:

$$T(G) \quad = \quad \begin{cases} tc(chain(G)) & \text{if } G \models \Psi_{\mathrm{c\&c}} \\ \langle X, \{(x,x) \mid x \in X\}\rangle & \text{if } G \models \neg\Psi_{\mathrm{c\&c}} \end{cases}$$

It is not hard to see that T is generic and PTIME-computable. Since T computes transitive closure of a chain, it is not first-order definable and hence not in $\mathcal{PR}(\mathrm{FO})$.

To show that $T \in \mathcal{WPC}(\mathrm{FO})$, we use the locality property and represent a sentence as a Boolean combination of Gaifman sentences α [15]:

$$\exists x_1 \ldots \exists x_s.(\psi^{(r)}(x_1) \wedge \ldots \wedge \psi^{(r)}(x_s) \wedge \bigwedge_{\substack{i,j \leq s \\ i \neq j}} d(x_i, x_j) > 2r)$$

Here $\psi^{(r)}(x)$ denotes a formula in which x is the only free variable and all quantifiers are of the form $\forall y \in N_k(x)$ or $\exists y \in N_k(x)$, where $k \leq r$ and $N_k(x)$ is the k-neighborhood of a node x. That is, $N_k(x)$ is the set of nodes that can be reached from x by a non-oriented path of length at most r. By $d(x,y) \geq i$ we denote the first-order fomula that expresses the fact that the distance between x and y is bigger than i.

Hence, it is enough to show that with respect to T, every such α has a weakest precondition. Since for every graph that does not satisfy $\Psi_{\mathrm{c\&c}}$ every neighborhood contains just one point, it is then enough to find preconditions assuming that the input graph G satisfies $\Psi_{\mathrm{c\&c}}$. This is done as follows. If $s > 1$ and $r \geq 1$, then α cannot be satisfied on a finite linear order; hence we take the precondition to be *false*. If $s \geq 1$ and $r = 0$, then we can take the precondition to be *false* if $\psi^{(r)}(x)$ fails in one-point neighborhoods where the only edge is the loop on that point. Otherwise $T(G) \models \alpha$ iff the chain part of G has at least s points, which can be tested by a first-order sentence.

If $s = 1$ and $r \geq 1$, then α becomes $\exists x.\psi^{(r)}(x)$. According to [16, 30], for every first-order sentence α', it is possible to find $n \in \mathbb{N}$ such that all linear orders of size $\geq n$ agree on α'. In fact $n = 2^{\mathrm{qr}(\alpha')}$. Hence, we can test if α is satisfied in linear orders of size up to $2^{\mathrm{qr}(\alpha)}$ and then construct the precondition as the corresponding Boolean combination of sentences about the cardinality of the chain part of G. Note that the above procedure for finding weakest preconditions is constructive in all cases. $\square$

The next result shows that the preconditions constructed by the algorithm in the proof of theorem 5 are rather complex.

Corollary 3 *Let T be the transaction from theorem 5. Then, for any $n > 1$, there exists a sentence α of quantifier rank n, such that if m is the quantifier rank of $\mathsf{wpc}(T, \alpha)$ computed by the algorithm of theorem 5, then $m \geq 2^n$.* $\square$

We do not know if a better algorithm is possible. In fact, it remains open whether polynomial time weakest precondition algorithms exist for any transaction in $\mathcal{WPC}(\mathrm{FO})$ that is not first-order definable.

The separation result continues to hold for more powerful logics, although it is impossible to find a separating transaction that will have such a nice description as above. First we observe the following.

Proposition 3 *For any Ω, $\mathcal{PR}(\mathrm{FO}_c(\Omega))$ can be captured by a transaction language.*

Proof. Since all functions and predicates in Ω are recursive, the set $\Gamma(D)$ is computable. The transactions in the language capturing $\mathcal{PR}(\mathrm{FO}_c(\Omega))$ are given by tuples $(\Gamma, \alpha_1, \ldots, \alpha_k)$ where Γ is a finite set of terms, and each α_i has n_i free variables. The generated transaction is defined to be such that the tuple $(\Gamma, \alpha_1, \ldots, \alpha_k)$ constitutes a prerelation. This fully determines the transaction, and the definition of prerelation implies that such a language captures $\mathcal{PR}(\mathrm{FO}_c(\Omega))$. $\square$

Sequential languages that define the same classes of transactions can be found in the literature, see for example [28, 29].

From proposition 3 and theorem 4 we obtain

Corollary 4 *For any Ω, there are transactions in $\mathcal{WPC}(\mathrm{FO}_c(\Omega)) - \mathcal{PR}(\mathrm{FO}_c(\Omega))$.* $\square$

However, in contrast to pure first-order logic, there are no generic separating transactions if constants are present in the language.

Proposition 4 *There are no generic transactions in $\mathcal{WPC}(\mathrm{FO}_c) - \mathcal{PR}(\mathrm{FO}_c)$.*

Proof sketch. Again assume the language of graphs, and let T be a generic transaction in $\mathcal{WPC}(\mathrm{FO}_c)$. Our goal is to show that T admits prerelations. Since T is generic, the set of nodes of $T(G)$ is always a subset of the set X of nodes of G. Thus, Γ can be taken to contain a single variable. To find a formula $\beta(x, y)$ such that $G \models \beta(a, b) \Leftrightarrow (a, b) \in T(G)$ for every $a, b \in X$, we let Ψ and Φ be $\mathsf{wpc}(T, E(c, d))$ and $\mathsf{wpc}(T, E(c, c))$ respectively, where c and d are arbitrary distinct elements of $\mathcal{U}$. Consider formulae $\psi(x, y)$ and $\phi(x)$ obtained from Ψ and Φ by replacing each occurrence of c by x and each occurrence of d by y. Define $\gamma(x, y)$ to be $(x = y \wedge \phi(x)) \vee (\neg(x = y) \wedge \psi(x, y))$.

Let C be the set of all constants that occur in γ. Then we can show that for any graph G whose set of nodes X is disjoint from C, $G \models \gamma(a, b)$ iff $(a, b) \in T(G)$ for every $a, b \in X$. We then construct a new formula $\beta(x, y)$ by replacing every atomic subformula of γ involving a constant from C by *false*. Using genericity, we can show that β defines prerelation for T. $\square$

5 Robustly verifiable transactions

Summarizing the results of the previous sections, we can say that many non-first-order transactions do not have weakest preconditions. Nevertheless, it is possible to find some non-first-order transactions that do have them. However, there is a gap between FO and FO_c. While we can find "nice" transactions that are not first-order definable and have weakest preconditions over FO, no such transactions exist when FO is replaced by FO_c. In particular, the $\mathrm{Datalog}^{\neg}$-definable transaction that provided the separation for $\mathcal{WPC}(\mathrm{FO})$ and $\mathcal{PR}(\mathrm{FO})$ no longer has preconditions if the language is FO_c. While this follows from proposition 4, we give a direct proof that illustrates the problems caused by the presence of constants.

Proposition 5 *The transaction $T \in \mathcal{WPC}(\mathrm{FO}) - \mathcal{PR}(\mathrm{FO})$ from the proof of theorem 5 is not in $\mathcal{WPC}(\mathrm{FO}_c)$.*

Proof. Let c be a constant symbol for some element of the universe. Assume that $T \in \mathcal{WPC}(\mathrm{FO}_c)$. Let $\alpha \equiv (\exists x \exists y. E(x, y) \wedge (x \neq y)) \wedge (\forall x. \neg E(x, c) \wedge \neg E(c, x))$ say that the graph has at least one edge which is not a loop, and c is not one of its nodes. Let $\beta = \mathsf{wpc}(T,\alpha)$ and $\gamma \equiv \beta \wedge (\exists x.E(x,c) \vee E(c,x))$. Then $G \models \gamma$ iff $G \models \Psi_{\mathrm{c\&c}}$ (since $T(G)$ has an edge which is not a loop), c is one of its nodes but c is not a node in $T(G)$. That is, G is a C&C-graph but not a chain. In other words, for the graphs in which one node is the interpretation of c, testing for chains would be FO_c-definable. However, this is impossible, and hence $T \notin \mathcal{WPC}(\mathrm{FO}_c)$. $\square$

Proposition 5 shows that a transaction verifiable over a language is not necessarily verifiable over even a simple extension of the language. This motivates the following definition. We call a transaction robustly verifiable it it is verifiable in every extension of the language. Formally, a transaction T is *robustly verifiable* over $\mathrm{FO}_c(\Omega)$ if for every extension of Ω to Ω' with a recursive collection of recursive predicates and functions, it is the case that $T \in \mathcal{WPC}(\mathrm{FO}_c(\Omega'))$. A transaction language $\mathcal{TL}$ is robustly verifiable if all its transactions are.

The following result, together with proposition 3, gives the proof of Theorem E. In what follows, we use $\mathcal{PR}(\mathrm{FO}_c(\Omega))$ to mean both the class of transactions that admit prerelations and the transaction language that captures this class. The existence of such a language was proved in proposition 3.

Theorem 6 *For every Ω, the transaction language $\mathcal{PR}(\mathrm{FO}_c(\Omega))$ is robustly verifiable over $\mathrm{FO}_c(\Omega)$, and every transaction robustly variable over $\mathrm{FO}_c(\Omega)$ is equivalent to a transaction in $\mathcal{PR}(\mathrm{FO}_c(\Omega))$.*

Proof sketch. As in the previous results, without loss of generality we will assume that the signature consists of a single binary relation. To show that transactions in $\mathcal{PR}(\mathrm{FO}_c(\Omega))$ are robustly verifiable, let T be given by a set of terms Γ and formula $\phi_E(x,y)$ giving the prerelation of $E(x,y)$ with respect to T. Without loss of generality, we can assume all terms in Γ have the same arity n. We give an algorithm WPC transforming every formula $\gamma(\vec{z})$ in $\mathrm{FO}_c(\Omega)$ into a formula $\mathrm{WPC}(\vec{z})$ such that $G \models \mathrm{WPC}(\vec{z}) \Leftrightarrow T(G) \models \gamma(\vec{z})$.

The algorithm is recursive in the logical complexity of γ. If γ is quantifier-free, WPC proceeds by replacing every occurrence of $E(\tau_1,\tau_2)$ where each τ_i is a term, by the formula

$(\exists \vec{y_1}. \bigvee_{\tau\in\Gamma} \tau(\vec{y_1}) = \tau_1(\vec{z})) \wedge (\exists \vec{y_2}. \bigvee_{\tau\in\Gamma} \tau(y_2) = \tau_2(\vec{z})) \wedge \phi_E(\tau_1(\vec{z},\tau_2(\vec{z}))$.

Then define

$$\begin{aligned} \mathrm{WPC}(\neg\gamma) &= \neg\mathrm{WPC}(\gamma) \\ \mathrm{WPC}(\gamma_1 * \gamma_2) &= \mathrm{WPC}(\gamma_1) * \mathrm{WPC}(\gamma_2),\ * \in \{\wedge,\vee\} \end{aligned}$$

If $\gamma \equiv \exists x \phi(x,\vec{z})$, define $\mathrm{WPC}(\gamma)$ to be

$$\begin{aligned} &\exists \vec{y_1}.(\textstyle\bigvee_{\tau\in\Gamma} \phi'(\tau(\vec{y_1}),\vec{z}) \ \wedge \\ &\quad \exists \vec{y_2}.(\textstyle\bigvee_{\rho\in\Gamma} \phi_E(\rho(\vec{y_2}),\tau(\vec{y_1})) \vee \phi_E(\rho(\vec{y_2}),\tau(\vec{y_1})))), \end{aligned}$$

where $\phi'(x,\vec{z}) = \mathrm{WPC}(\phi)$. It is routine to verify that the algorithm WPC is correct.

For the other direction, we will prove that any transaction T_0 that is not in $\mathcal{PR}(\mathrm{FO}_c(\Omega))$ is not robustly verifiable. Fix such a T_0. Clearly, we can also assume that T_0 is in $\mathcal{WPC}(\mathrm{FO}_c(\Omega)$. Let $\underline{\mathrm{P}}$ be an additional 4-place predicate symbol. Let L^+ be the set of all first-order formulae in the language with symbols for all constants in the universe, all elements of Ω, and the symbol $\underline{\mathrm{P}}$.

Our proof will proceed by inductively constructing a recursive interpretation P of $\underline{\mathrm{P}}$ such that T_0 does not admit definable weakest preconditions over the extension of $\mathrm{FO}_c(\Omega)$ by P. Our attempt to construct such a 'bad' P will be by diagonalization: we will pick a certain sentence ψ, and for each possible sentence S we will pick an initial segment of P designed to prevent S from being the precondition of ψ. Using the fact that T_0 is not in $\mathcal{PR}(\mathrm{FO}_c(\Omega)$, we can show that this diagonalization succeeds, which implies that T is not robustly verifiable.

Let *P-maximal*(s,t) be the L^+ formula $(E(s,t) \wedge \exists x\exists y.\ \underline{\mathrm{P}}(x,y,s,t)) \wedge (\forall x\forall y.\ (E(x,y) \rightarrow \neg\underline{\mathrm{P}}(s,t,x,y)))$. Let

$$\psi \ \equiv\ \exists s\exists t.\ P\text{-}maximal(s,t)$$

That is, ψ is the L^+-sentence asserting that there is a P-maximal pair.

If p and q are partial functions, then $q \succ p$ means that q extends p as a function. For any set A, let χ_A be its characteristic function.

Then we have:

Lemma 7 *For each partial function p, for each transaction T in $\mathcal{WPC}(\mathrm{FO}_c(\Omega) \setminus \mathcal{PR}(\mathrm{FO}_c(\Omega))$, there is a partial function p' and graph G such that for all finite P with $\chi_P \succ p'$,*

$$(*\phi) \qquad \neg(\langle G,P\rangle \models \phi \Leftrightarrow \langle T(G),P\rangle \models \psi)$$

Furthermore p' can be constructed effectively from p.

Given this lemma, the proof is a straightforward inductive construction: Let $\langle\phi_i\rangle$ be an enumeration of all the sentences in L^+, and let p_0 be empty, and p_{n+1} be any finite $p' \succ p_n$ such that p' guarentees $(*\phi_n)$. We can further ensure that $\bigcup_n p_n$ defines a total function by throwing the nth element of the universe $\mathcal{U}$ into the domain of p_n if it is not there already.

Let P be the unique set such that $\chi_P \succ p_n$ for all n. We can ensure P is recursive, since the construction of p_n can be done recursively. It is now easy to see that for P as above, there can be no sentence of L^+ that holds in a graph G exactly when $\langle T(G), P\rangle \models \psi$. Hence T does not admit weakest preconditions over L^+, which shows T is not robustly verifiable. $\square$

Corollary 5 *$\mathcal{PR}(\mathrm{FO}_c(\Omega))$ is the maximal robustly verifiable language over $\mathrm{FO}_c(\Omega)$.* $\square$

6 Concluding remarks

In this paper we have looked at the problem of verifying transaction safety *before* transactions are executed. The main results can be summarized as follows. If integrity constraints are specified in first-order logic, then for first-order transaction languages it is possible to compute both weakest preconditions and prerelations. However, if a transaction language has a mechanism for doing recursion, then such ability is generally lost. There are still some transactions that are not first-order definable but have weakest preconditions. However, if we require that the ability to calculate weakest preconditions be independent of extensions to the language, then we can use only languages that admit prerelations.

The last statement, which is a reformulation of corollary 5 can be interpreted as follows. If we are interested in designing a "nice" transaction language that is verifiable over FO_c, and the verification algorithm can be extended to encompass additions to the signature, then we cannot hope that the language will be more expressive than the first-order transaction language defined in [28, 29].

In this paper we only discussed whether it is possible to compute preconditions or prerelations, and what are the implications of our ability to compute them. The algorithmic aspects of computing preconditions were left unexplored. In particular, we would like to address in our future work the problem of *efficiently* computing preconditions. We believe it is of practical importance to identify fragments of transaction languages such as those in [3, 4, 28] for which computing preconditions can be done efficiently (say, in polynomial time).

Computing preconditions may depend on a form in which transactions are specified. For simple relational transactions it is possible to reason about their equivalence [22]. Combining this reasoning together with the theorem proving approach of [31] is an interesting direction to pursue. That is, first we may try to find a better analyzable transaction which is equivalent to the original one, and then try to test its safety.

As mentioned in the introduction, we are interested in transforming a verifiable transaction T into a safe transaction <u>if</u> $\mathsf{wpc}(T, \alpha)$ <u>then</u> T <u>else</u> *abort* which will maintain α as an invariant. As pointed out in [27, 17, 18, 25, 26], assuming that α is always true, it may be possible to find a Δ, which is much simpler than $\mathsf{wpc}(T, \alpha)$, such that $\alpha \rightarrow (\Delta \leftrightarrow \mathsf{wpc}(T, \alpha))$. Using this we can transform T to <u>if</u> Δ <u>then</u> T <u>else</u> *abort* which is more efficient. We are interested in studying classes of transactions for which such a simplification can be efficiently carried out.

Acknowledgements: We would like to thank Neil Immerman and Scott Weinstein for their help.

References

[1] S. Abiteboul, R. Hull and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1994.

[2] S. Abiteboul and V. Vianu. Transactions and integrity constraints. In *Proceedings of the 4th Symposium on Principles of Database Systems*, 1985, pages 193–204.

[3] S. Abiteboul and V. Vianu. Procedural languages for database queries and updates. *Journal of Computer and System Sciences* 41 (1990), 181–229.

[4] S. Abiteboul and V. Vianu. Datalog extensions for database queries and updates. *Journal of Computer and System Sciences* 43 (1991), 62–124.

[5] M. Ajtai and R. Fagin. Connectivity is harder for directed than for undirected graphs. *Journal of Symbolic Logic* 55 (1990), 113–150.

[6] R.J.R. Back. Proving total correctness of nondeterministic programs in infinitary logic. *Acta Informatica* 15 (1981), 233–249.

[7] M. Benedikt, G. Dong, L. Libkin and L. Wong. Relational expressive power of constraint query languages. This volume.

[8] M. Benedikt, T. Griffin and L. Libkin. Verifiable properties of database transactions. Bell Labs Technical Memo, 1995.

[9] R. Berghammer, B. Elbl and U. Schmerl. Formalizing Dijkstra's predicate transformer wp in weak second-order logic. *Theoret. Comp. Sci.* 146 (1995), 185–197.

[10] V. Breazu-Tannen and R. Subrahmanyam. Logical and computational aspects of programming with sets/bags/lists. In *LNCS 510: Proc. of 18th ICALP, Madrid, Spain, July 1991*, pages 60–75. Springer Verlag, 1991.

[11] E. W. Dijkstra. Guarded commands, nondeterminacy and formal derivations of programs. *Comm. ACM* 18 (1975), 453–457.

[12] E. W. Dijkstra. *A Discipline of Programming.* Prentice-Hall, 1976

[13] R. Fagin. Finite model theory – a personal perspective. *Theoretical Computer Science* 116 (1993), 3–31.

[14] R. Fagin, L. Stockmeyer, and M. Vardi. On monadic NP vs. monadic co-NP. *Information and Computation* 120 (1995), 78–92.

[15] H. Gaifman. On local and nonlocal properties. In J. Stern, editor, *Logic Colloquium '81*, pages 105–135. North-Holland, 1982.

[16] Y. Gurevich. Toward logic tailored for computational complexity. In *Proccedings of Computation and Proof Theory*, Springer Lecture Notes in Mathematics, vol. 1104, 1984, pages 175–216.

[17] L. J. Henschen, W. W. McCune, and S. A. Naqvi. Compiling constraint-checking programs from first-order formulas. In H. Gallaire, J. Minker, and J. Nicolas, editors, *Advances in Database Theory*, pages 145–170. Plenum Press, 1984.

[18] A. Hsu and T. Imielinski. Integrity checking for multiple updates. In *Proceedings of ACM-SIGMOD 1985 International Conference on Management of Data*, pages 152–168, 1985.

[19] K. Etessami. Counting quantifiers, successor relations, and logarithmic space. In *Proceedings of 10th IEEE Conference on Structure in Complexity Theory*, May 1995, pages 2–11.

[20] N. Immerman. Languages that capture complexity classes. *SIAM J. Comput.* 16 (1987), 760–778.

[21] N. Immerman, S. Patnaik and D. Stemple. The expressiveness of a family of finite set languages. In *Proceedings of the 10th Symposium on Principles of Database Systems*, 1991, pages 37–52.

[22] D. Karabeg and V. Vianu. Simplification rules and complete axiomatization for relational update transactions. *ACM Trans. Database Systems* 16 (1991), 439–475.

[23] L. Libkin and L. Wong. New techniques for studying set languages, bag languages and aggregate functions. In *Proceedings of the 13th Symposium on Principles of Database Systems,* Minneapolis MN, May 1994, pages 155–166.

[24] L. Libkin and L. Wong. Some properties of query languages for bags. In *Proceedings of the Fourth Workshop on Database Programming Languages, Manhattan NY, August 30–September 1, 1993*, Springer Verlag, 1994, pages 97–114.

[25] W. W. McCune and L. J. Henschen. Maintaining state constraints in relational databases: A proof theoretic basis. *Journal of the ACM* 36 (1989), 46–68.

[26] J.-M. Nicolas. Logic for improving integrity checking in relational data bases. *Acta Informatica*, 18:227–253, 1982.

[27] X. Qian. An effective method for integrity constraint simplification. In *Fourth International Conference on Data Engineering*, 1988.

[28] X. Qian. An axiom system for database transactions. *Information Processing Letters* 36 (1990), 183–189.

[29] X. Qian. The expressive power of the bounded-iteration construct. *Acta Informatica* 28(7):631–656, October 1991.

[30] J. G. Rosenstein. *Linear Orderings.* Academic Press, New York, 1982.

[31] T. Sheard and D. Stemple. Automatic verification of database transaction safety. *ACM Trans. Database Systems* 14 (1989), 322–368.

[32] D. Stemple, S. Mazumdar and T. Sheard. On the modes and meaning of feedback to transaction designers. In *Proceedings of ACM-SIGMOD 1987 International Conference on Management of Data*, pages 374–386, 1987.